Navigating Compliance with Data Transfers in Federated Data Processing

Kaustubh Beedkar
TU Berlin
kaustubh.beedkar@tu-berlin.de

Jorge Quiané
TU Berlin & DFKI
jorge.quiane@tu-berlin.de

Volker Markl
TU Berlin & DFKI
volker.markl@tu-berlin.de

Abstract

In an increasingly digital world, compliance with data regulations play an important role. More and more individuals are rapidly getting concerned with the way their data is being stored and processed by organizations. Therefore, it is crucial that data processing be subjected to regulatory obligations at its core. Yet, achieving compliance with data regulations requires the entire data processing pipeline to be revisited to embrace data policies as first-class citizens. In this paper, we present our work on novel systems and methods for federated data processing, where the processing of geo-distributed data is subjected to data transfer regulations. We showcase our work on compliant geo-distributed data processing and present research challenges and opportunities for a federated data processing system to make compliance truly its first-class citizens.

1 Introduction

Federated data processing has been a standard model for virtual integration of disparate data sources, where each source upholds a certain amount of autonomy. While early federated technologies resulted from mergers, acquisitions, and specialized corporate applications, recent demand for decentralized data storage and computation in information marketplaces[32]) and for geo-distributed data analytics [33, 22, 14] has made federated data services an indispensable component in the database market. Cloud providers such as AWS, Google, and Microsoft have also adopted distributed query capabilities within their products to support federated data processing.

Running analytics in a federated environment mainly relies on distributed query processing frameworks, such as those based on data integration systems (e.g., [25]) and/or multi-database systems (e.g.,[3, 31]). At high-level, a distributed query processing framework provides a unified query interface to query distributed and decentralized data. It transparently translates a user-specified query into a so-called query execution plan. To do so, a query optimizer considers distributed execution strategies (involving distributing query operators like join or aggregation across compute nodes), communication cost between compute nodes, and introduces a global property that describes where, i.e., at which site, processing of each plan operator happens. For example, a two-way join query over data sources in Asia, Europe, and North America may be executed by first joining data in North America and Europe and then joining with the data in Asia. As one can notice, federated queries implicitly ship data (i.e., intermediate query results) between compute sites. While several performance aspects, such as

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

bandwidth, latency, communication cost, and compute capabilities have received great attention, the federate nature of data processing has been recently challenged by data transfer regulations (or policies) that restrict the movement of data across geographical (or institutional) borders or by any other rule of data protection that may apply to the data being transferred between certain sites. European directives, for example, regulate transferring only certain information fields (or combinations thereof), such as non-personal information or information not relatable to a person. Likewise, regulations in Asia may also impose restrictions on data transfer. Non-compliance to such regulatory obligations has attracted fines in the tune of billions of dollars[10]. It is, therefore, crucial to consider compliance with respect to legal aspects when analyzing federated data.

Nevertheless, complying with regulations when transferring data is a big challenge today. One has to expand the capabilities of modern federated data processing systems to aid data controllers (i.e., entities that determine what data and how the data should be processed) and data processors (i.e., entities that provide data storage and processing capabilities) in navigating compliance with data transfers. In particular, we need (a) a declarative language for expressing data transfer rules, (b) to revisit query rewriting and optimization techniques to translate user queries transparently into compliant query execution plans, and (c) to revisit query execution to support decentralized data processing across heterogeneous compute nodes.

In this paper, we outline the main system challenges required for complying with data transfer obligations in the context of federated data processing. We showcase our work on compliant data processing, which offers limited capabilities in navigating compliance. We then discuss our current research endeavors that overcome prior limitations and discuss open problems and research challenges.

2 Problem Scope & Challenges

We start by giving a birds-eye view of federated data processing systems (FDPS) and then outline aspects of data regulations that affect the transfer of data between national (or institutional) borders. We then discuss the challenges that we set out to address in order to expand the capabilities of FPDS to navigate compliance with data transfers.

2.1 A Brief Recall on Federated Data Processing

An FDPS consists of three major components as illustrated in Figure 1: a data interface, a query optimizer, and a query processor. The data interface provides end-users (e.g., data analysts or data administrators) the ability to query and process data that is stored across distributed data stores in a unified manner. The data interface upon receiving a user query (e.g., SQL) parses and translates it into a framework-specific internal structure (e.g., a logical query plan). The query optimizer then rewrites the logical query plan into a query execution plan (QEP). It extends a single-site data processing across distributed compute notes. To do so, it considers communication costs between compute nodes and introduces a global property that describes where, i.e., at which site processing of each plan operator happens. The query processor then "orchestrates" the actual execution of the query, which results in the transfer of (intermediate) query data between compute nodes. In geo-distributed environments, compute nodes are located across national (or institutional) borders. In this context, the transfer of data between sites may result in non-compliance to data transfer regulations.

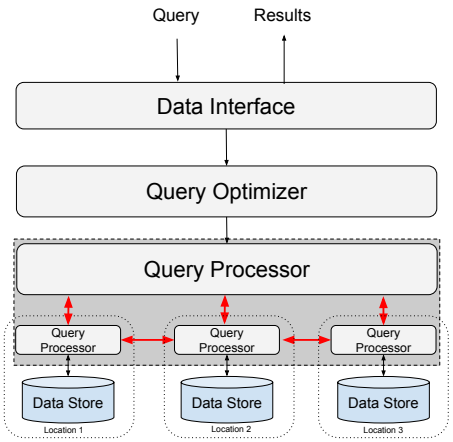


Figure 1: Federated Data Processing

2.2 Primer on Data Transfer Regulations

Data regulations, such as EU's General Data Protection Regulation (GDPR) [1] or California Consumer Privacy Act (CCPA) [2] significantly affect how data is stored, processed, and transfer. In this section, we aim to understand regulations from the perspective of data transfers. To achieve that, we analyze GDPR articles that regulate the transfer of data across national borders[1].

GDPR articles 44–50 explicitly deal with the transfer of data across national borders. Among these, we identified two articles and one recital wherein the legal requirements for transferring data fundamentally affect FDPS components.

Article 45: Transfers on the basis of an adequacy decision. The article dictates that transfer of data may take place without any specific authorization, e.g., when there is adequate data protection at the site where data is being transferred or when data is not subjected to regulations (i.e., when the data does not follow the definition of personal data as in Article 4(1)).

Article 46: Transfers subject to appropriate safeguards. This article prescribes that (in the absence of applicability of Article 45) data transfer can take place under "appropriate safeguards". Based on the European Data Protection Board (EDPB) recommendations that supplement transfer tools, *pseudonymisation* of data (as defined under Article 4(5)) is considered as an effective supplementary method.

Recital 108: Transfers under measures that compensate lack of data protection. Data after adequate *anonymization* (i.e., when resulting data does not fall under Article 4(1) and as described in Recital 26) does not fall under the ambit of GDPR and therefore can be transferred.

Discussion. Based on the above regulations, we observe that depending on the data and to where that data is being transferred, we can classify data transfer regulations into:

- *No restrictions on transfer.* Some data maybe allowed to be transferred unconditionally, and some to only certain locations.
- *Conditional restrictions on transfer*. For some data, only derived information (such as aggregates) or only after anonymization, can be transferred to (certain) locations.
- *Complete ban on transfer*. Some data, no matter whatsoever, must not be transferred outside.

2.3 Compliance by Design: Research Challenges

Our overreaching goal is to develop methods and systems that aid *data controllers* (entities that control what data and how the data should be processed) and *data processors* (entities that processes data on behalf of a controller) by providing appropriate safeguards within FDPS such that transfer—as a result of federated data processing—of data across borders complies to regulatory obligations described above.

Declarative Data Transfer Rules. The first and foremost challenge to achieve compliance by design is to have declarative languages for specifying data transfer regulations. Doing so is not trivial as regulations affect data differently depending on its type, it's processing, and the location where it is processed (or transferred). For example, regulations may apply to an entire dataset, parts of it, or even information derived from it. Furthermore, datasets are heterogeneous in their data models (e.g., graphs, relational, and textual). Therefore, devising a declarative language where one can specify data constraints in an easy and effective manner is far from being simple.

[1] We note that compliance to GDPR aspects, such as collecting, securing, storing, deleting are beyond the scope of our current focus (see discussion in Section 5).

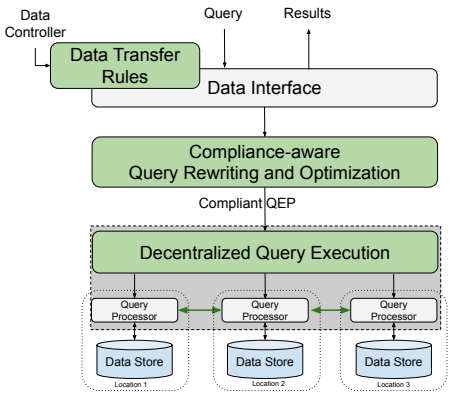


Figure 2: Compliant FDP

Compliance-aware Query Rewriting and Optimization Once a user specifies the data policies on her data, she then needs effective and efficient ways to process federated queries in a manner that the processing is compliant. Achieving so is challenging as we need to extend query rewriting and optimization capabilities. A query optimizer should be able to transparently translate logical query plans into compliant query execution plans by "injecting" operations that transform data such that the transformed data prescribes to regulations affecting its transfer.

Decentralized Query Execution Lastly, we also need to revisit query processors than can execute a compliant query plan across distributed (potentially heterogeneous) compute nodes. In contrast to current approaches that employ a mediator-based execution, we need meta-execution engines, which can delegate query execution to disparate compute nodes and de-centrally execute the query, without involving itself in the query execution. Achieving so is crucial to make sure that data processing happens on the locations prescribed by the query optimizer, irrespective of the location of the processor.

3 Compliance-aware Query Rewriting and Optimization

We start by giving the overall idea of our approach for achieving compliant data processing. We then discuss our research on supporting compliance assuming relational workloads before outlining open problems and current research directions going beyond the relational world.

3.1 Overall Idea

A crucial aspect in adhering to data transfer regulations is to transform the data in a way that renders it suitable to be transferred across borders. From the perspective of a FDPS, transforming (intermediate) data before shipping it to another compute location can be considered as performing additional *masking operations* on the data. To illustrate this, consider a two-way join query that access data stored across Europe, North America, and Asia. Figure 3 (left) illustrates a logical query plan for this query where orange boxes denote cross-border operations (e.g., join operators) that require inputs from two or more sources and blue circles denote other query operators (e.g., map, filter, or aggregate). On the right, we illustrate a corresponding execution plan, where the optimizer decides at which site the processing of each plan operator should happen (e.g., both cross-border operators must be performed in North America). The optimizer also rewrites the query by reordering the query operators (e.g., selection pushdowns) and by "injecting" data masking operators (shown by red boxes) as a means to provide appropriate safeguards for cross-border data transfers. In this example, both cross-border operations happen in North America, and data from EU and Asia is transformed by a masking operator before being shipped to North America.

3.2 Navigating Compliance in the Relational Paradigm

In our current approach, we confine to processing of data that is stored in geo-distributed SQL databases and propose query rewriting and optimization techniques that preserve the query semantics. In more detail, we focus on data transfer rules that can be adhered to by data masking via relational operations (e.g., project, aggregate, or filter) such that the resulting compliant QEP retains the query semantics, i.e., the output of the query should be the same as if there were no data transfer constraints. For instance, a projection operator can mask certain

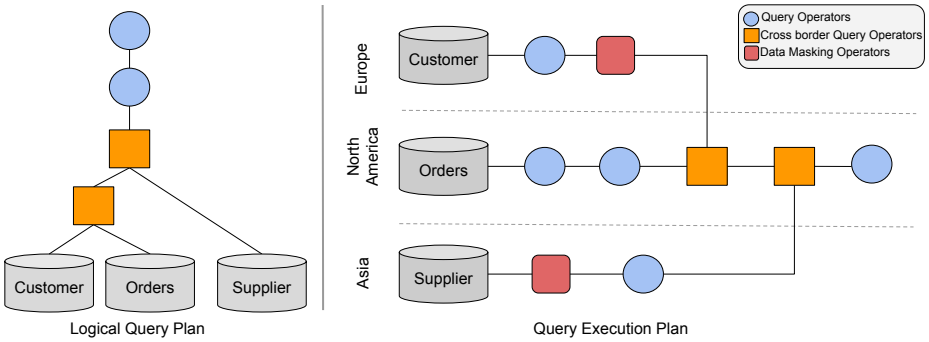


Figure 3: Illustration of Query Rewriting and Optimization: (left) Logical query specified by user and (right) execution plan derived by the query optimizer after injecting masking operators.

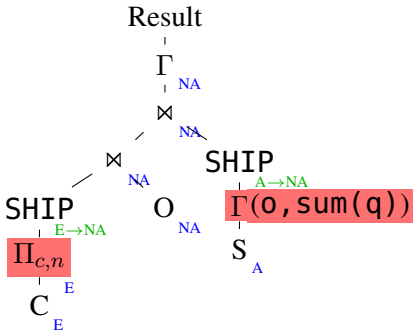
columns by projecting them out before the (intermediate data) is transferred to another location, and when the masked columns are not required by the query later.

Let us illustrate this approach by expanding upon the previous example. Consider the following schema for the Customer, Orders, and Supplier tables along with data transfer rules that apply to data at each location.

`Customer`	(**c**ustkey, **n**ame, **a**cctbal, **m**ktseg, **r**egion)	Customer data can be transferred outside only after suppressing account balance information
`Orders`	(**c**ustkey, **o**rdkey, **t**otprice)	Only aggregated Orders data can be transferred to Asia and an order's price cannot be transferred to Europe
`Supply`	(**o**rdkey, **q**uantity, **e**xtprice)	Only aggregated Supply data for orders' quantity and extended price from Asia can be transferred to North America.

Furthermore, consider a query Q_{ex}

```
SELECT C.name, SUM(O.totprice), SUM(S.quantity)
FROM Customer AS C, Orders AS O, Supply AS S
WHERE C.custkey=O.custkey AND O.ordkey=S.ordkey
GROUP BY C.name
```



The query plan on the right shows a compliant QEP as derived by our query optimizer (discussed below). Here the `SHIP` operator describes the point where intermediate results are communicated between two sites and Γ denotes the aggregation operator. Annotations corresponding to each operator describe where processing of the plan operator should happen. Observe, that executing such a plan will not violate any of the above rules: it performs both join operations in North America, masking data via the projection operator $\Pi_{c,n}$ suppresses the account balance information of Customers (before the data is shipped from Europe to North America) and via the aggregation operator Γ(`o,sum(q)`) suppresses the orders' quantity (prior to shipping data from Asia to North America), as desired by the rules.

Policy Expression Language. One of the challenges in automatically translating user-specified queries into compliant QEPs is to first integrate rules into the query optimization framework. For this, we have developed a policy expression language that provides a simple and intuitive (SQL-like) syntax to specify *which* and *where*

data are allowed to be transferred. We basically define two kinds of policy expressions: basic and aggregate expressions. A basic expression is of the form of a Select-Project query that can specify restrictions pertaining to certain tables, rows, and/or columns. An aggregate expression is of the form of a Select-Project-GroupBy query and further allows specifying restrictions pertaining to the transfer of aggregated information.

Basic Expressions. Basic expressions allow specifying shipping of certain rows and columns of a table to another location and have the following syntax:

```
ship attribute list from table to location list where condition list
```

This expression specifies cells, i.e., rows and columns, of a table to be transferred without affecting the query semantics.[2] The specified cells from the table in the **from** clause (i) belong to both columns in the **ship** clause and tuples that satisfy the predicates in the **where** clause, and (ii) can be transferred to locations in the **to** clause. Intuitively, if a subquery accesses only the specified cells, then its output can be transferred to locations specified in the expression. Consider the data transfer rule from the above example, which does not allow for shipping the account balance information of customers outside Europe. Suppose the rule also allowed for shipping customer's mktsegment and region information to North America for commercial customers. We can use the following two policy expressions:

```
ship custkey,name from Customer C to Asia, North America
ship mktseg, region from Customer C to North America where mktseg='commercial'
```

Aggregate Expressions. For certain data, transfer rules only allow shipping of aggregated information. For these cases, we have aggregate expressions that allow us to specify aggregations over columns. The syntax of an aggregate expression is given as:

```
ship attribute list as aggregates aggregate types from table to location list
where condition list group by attribute list
```

In the above syntax, the list of attributes in the **ship** clause specifies cells of columns that should be aggregated before being transferred to locations in the location list. The **as aggregate** clause specifies aggregation functions that should be used to aggregate specified cells. As before, the specified cells must belong to columns in the attribute list for the tuples that satisfy the predicate in its **where** clause. Lastly, the **group by** clause specifies lists of grouping attributes for which the specified cells can be grouped by zero, one, or more attributes from its attribute list. Consider again the Customer data from the above example and assume that account balance information can be transferred only after aggregating. A possible expression is:

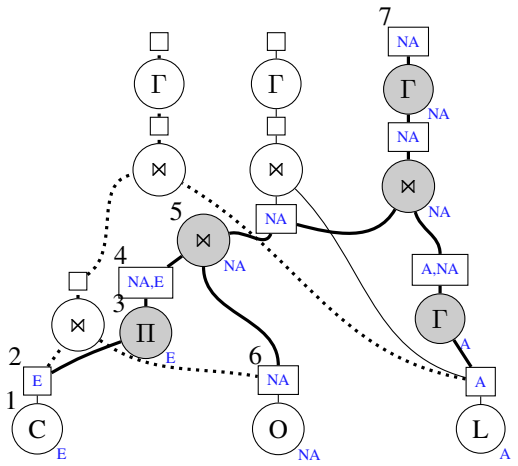
```
ship acctbal as aggregates sum, avg from Customer C to * group by mktseg, region
```

The above expression specifies how values of the acctbal column of the Customer table can be transferred outside. In particular, it specifies that (i) acctbal should be aggregated via the functions SUM or AVG and (ii) the cells of the acctbal column can be grouped by mktsegment and/or by nationkey. For example, output of the queries $\mathcal{G}_{\text{sum}(acctbal)}(C)$ and ${}_{region}\mathcal{G}_{\text{avg}(acctbal)}(C)$ can be transferred to all locations, whereas of $\mathcal{G}_{\text{sum}(acctbal)}(\sigma_{name='abc'}(C))$ and $\Pi_{acctbal}(C)$ cannot be transferred at all.

Compliance-based Optimizer. Now, in the query optimization phase, our optimizer aims to determine if a query is legal (i.e., its execution does not lead to violating data transfer rules) and to automatically generate an optimal compliant plan. We follow a two-phase optimization process that comprises plan annotation and site selection. The plan annotator receives a logical plan as input and outputs an annotated QEP. An *annotated QEP* is an optimized logical plan in which each plan operator is annotated with a set of compliant sites (i.e., sites where the execution of the operator will not violate any dataflow constraint). The site selector then uses dynamic programming to find the optimal placement of query plan operators taking data shipping cost into account.

[2]For exposition, we restrict to expressions over a single table. This is not a limitation: one can specify a policy expression over more than one base table. In this case, the condition list in the **where** clause of the expression must contain the join predicate.

More specifically, we adapt the Volcano optimizer generator [12] to generate the plan annotator. Our adaptations allow us to produce an annotated plan by enumerating the plan space by applying algebraic equivalence rules in a top-down fashion and filter compliant ones by applying our annotation rules in a bottom-up fashion. To do so, we treat geo-locations associated with base tables as "interesting properties" and propagate these properties bottom-up via annotation rules. Our annotation rules are based on the structure of the subplans and make use of a lightweight mechanism to evaluate data transfer rules. Our policy evaluator allows for easy integration of policy expressions into the annotation process. In particular, during plan enumeration, it determines to which cross-borders sites the output of operators can be shipped. The figure on the right illustrates the annotation process for our running example. Here the plan with the dotted lines shows the initial logical plan. The plan with thick solid lines shows the annotated plan which the annotator outputs. The letters in the square boxes denote sites to which the output of an operator can be shipped to and letters below each operator denote sites where each plan operator can be executed. For example, the project operator (node 3) must be executed in Europe but its output (node 4) can be shipped to North America and Europe. It is easy to see that the plan with thick solid lines translates to the compliant plan shown above. For a more detailed description of our optimizer, we refer readers to [6], which also gives a more formal treatment and proof of correctness.



3.3 Compliance Beyond the Relational Paradigm

Data masking via relational operations and our above query rewriting and optimization techniques have inherent limitations. They limit the whole gamut of compliant QEPs. As conditional restrictions on data transfers (as discussed in Section 2.2) may allow transferring of pseudonymized and/or anonymized data, it is important to expand the scope of masking functions to beyond relational operations.

Advanced Masking Functions. Based on European Commission's Opinions on Anonymization Techniques [21], we consider the following masking operations in addition[3].

Masking function	*Description*
Suppression	*Similar to Projecting out, replaces a value with a generic value (e.g., 'xxx')*
Pseudonymisation	*Replaces one value with another, s.t., new value has no logical relationship to the original (e.g., 'abc' to 'xyz')*
Blurring	*Alters a value by partial suppression (e.g., 'abc' → 'aXX')*
Generalization	*Generalizes a value using a predefined domain hierarchy (e.g., '23' → '20–25')*
Shuffling	*Replaces existing values with values from the same column*
Noise Addition	*Alter accuracy of (numeric) attributes*

Interleaving masking operators with query operators. A direct consequence of masking via non-relational operations is that: (1) it may no longer be possible to preserve query semantics. To illustrate this, consider that when transferring employee records, the age attribute is generalized by a function f that translates numeric attributes to certain range intervals (e.g., $f(23) = 20 - 25$). Such a masking may change the data type (e.g., `int`

[3]We note that this is not an exhaustive list of masking functions that we plan to support

to `int4range`) leading to change in query semantics. As another example, now consider that the age attribute is masked via suppression. In this case too, the resulting records will contain fewer attributes than that desired by the query. (2) Naive interleaving may affect robustness of the data masking. For example, consider a masking function f that blurs zipcode (e.g., $f(12345) = 123xx$). In this case a filter predicate p on zipcode (e.g., $p \equiv$ zipcode=12345) evaluated before masking may lead to possible singling out of an individual records. To this end, our current work explores the following research questions.

Q1 *How to interpose masking functions with query operators such that the resulting data is still anonymized?* For example, for scalar and univariate masking functions (e.g., blurring), we can substitute filter predicate by an UDF filter where the UDF is the masking function (e.g., we can rewrite p as zipcode=$f(12345)$.

Q2 *How to minimize information loss by "injecting" the right masking functions?* For example, by exploiting the fact that noise addition preserves aggregates (such as SUM() and AVG()), we can use masking via noise addition for aggregate queries instead of masking by blurring.

General Purpose Dataflow Programs. Many data analytics tasks are expressed as directed (a)cyclic graphs (DAG) composed of second order functions (e.g., map). To this end, we are investigating how advanced masking function can be composed with dataflow operators. For this, we plan to leverage our prior work on dataflow optimizations [13] and investigate effective and efficient ways to support (iterative) DAG programs.

4 Decentralized Query Execution

We now turn our attention to the execution of compliant QEPs. We first discuss why current FDPS fall short of executing compliant QEPs. We then present key challenges we need to tackle before presenting our approach.

State-of-the-art & Limitations. State-of-the-art FDPS (such as Presto [25]) mostly follow a mediator-based approach [16]. Although, such an approach (as illustrated by the figure on the right) is useful for performing data analytics across heterogeneous compute nodes, it does not lend itself to executing compliant QEPs. This is because *cross-database operations* (i.e., operators requiring inputs from multiple databases) are performed by the mediator's execution engine. This leads to the added complexity in ensuring compliance (due to the centralized processing by the mediator) to data transfers during query execution. To see this, recall the example of Section 3.2, and consider that now Orders data from North America can be transferred to Asia but nothing can move out from Asia. Under these constraints, a possible compliant plan is to first ship the data (after masking via projection operator as before) from Europe to North America, perform the first cross-database operation ($R \equiv C \bowtie O$) in Europe, and the second cross-database operation ($R \bowtie S$) in Asia. Current FDPS cannot execute such multi-site queries. Another limitation arises when metadata (or statistics) cannot be freely shared across sites, which leads to bad QEPs.

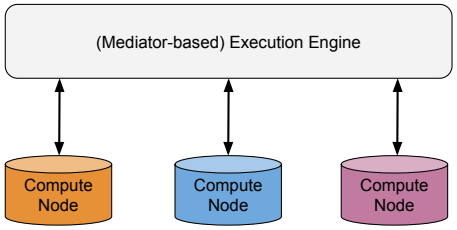


Figure 4: Centralized FDP

Challenges. In contrast to mediator-based approaches, we need a fully-decentralized approach (i.e., without any central entity in the execution pipeline) to execute queries over geo-distributed compute nodes. This, however, poses several challenges:

1. System Interoperability: A key aspect in achieving a fully-decentralized query execution is the ability to communicate (intermediate) data between underlying data processing systems. A key challenge, therefore, is to make systems interoperable without affecting their autonomy, even without them noticing.

2. Fast Data Transfer: As a consequence of lack of systems' interoperability, moving data among different data processing systems incur a high cost (e.g., we might need to export data from one DBMS and import that data in another DBMS). The challenge resides in enabling "native" data transfers (e.g., reading a relation from a DBMS directly from its binary store) without affecting the autonomy of the underlying processing systems.

3. Incompatible Data Formats: Different systems work on different internal data formats, which adds additional complexity and non-trivial cost of storing and converting data from one format to another during data transfers. The main question to answer is: does an intermediate data representation exists that can speed up the conversion from any source data format to any target data format?

4. Query Optimization: Last but not the least, optimizing queries without having a "centralized" access to data systems makes query optimization across geo-distributed data processing systems non-trivial. For example, data processing systems have their own cost models, which impedes in determining global cost of executing queries.

Our Approach. We have developed a meta-execution engine, which enables executing federated queries in a decentralized fashion. Figure 5 illustrates our overall approach. In contrast to a mediator-based approach, we follow what we refer to as a delegator-based approach. The Meta-execution engine, delegates the entire query execution to underlying data processing systems. This avoids a need to have any central entity in the execution pipeline, and data transfer only happens among compute nodes following annotations as prescribed by the compliance-aware query optimizer.

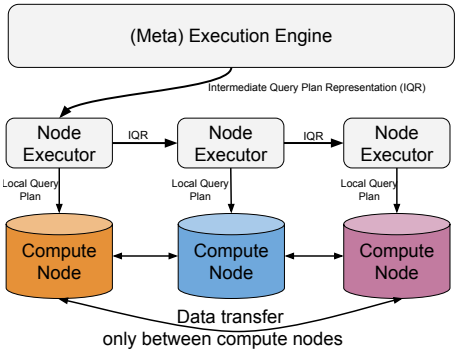


Figure 5: Decentralized FDP

In more detail, the meta execution engine first globally optimizes a query (with limited available metadata) and translates the plan into an intermediate representation (IQR). The IQR is agnostic to underlying data processing platforms and encapsulates query semantics comprising local subplans (i.e., processing that should be limited to certain locations) and inter-operator (cross-database) communication endpoints as well as mechanics of data movement between different systems. The node executors are local meta execution engines that are responsible for (a) translating and optimizing a local-subplan into platform specific query plan (e.g., to a Spark program), (b) communicating the IQR to other (relevant) node executors, and (c) facilitating the data movement between systems either by leveraging underlying system's capabilities (see below) or by creating suitable data "pipes". It is important to note that the node executors themselves do not execute any part of the query but only delegate execution to underlying systems.

In our current prototype [5], we support such decentralized query execution over multiple RDBMSes (including PostgreSql, MariaDB, MySQL, Hive, and DB2). To do so, the local node executors leverage the SQL/MED standard to communicate intermediate query data between systems. More specifically, while translating into a local DBMS specific program, the node executor first registers external tables (i.e., tables corresponding to the output of remote subplans) as local tables. This enables achieving a completely decentralized query execution.

Current Research Directions. We are currently investigating how to process data using geo-distributed heterogeneous compute platforms. For example, how can we execute a multi-site query using a PostgreSql database at one site and a Spark Cluster at another site? This is crucial to support compliance for non-relational workloads. To this end, our work includes expanding capabilities of cross-platform data processing systems (such as Apache Wayang [3]) to support geo-distributed compute sites. We are also investigating suitable common data formats such as Parquet, Protocol Buffers, or Avro, which makes inter-system data transfers efficient.

5 Related Work

We now relate the ideas presented in this paper to prior work on compliance and federated data processing.

Compliance. With growing concerns over data privacy and enforcement of regulations, several works have looked into various legal contexts within with data processing systems must be designed. With respect to GDPR compliance, most works have focused on data subjects' rights as a large proportion of GDPR governs data storage. In this regard, [20] analyzed various aspects of GDPR including deletion, indexing, monitoring and logging, and access control, and how they impact database systems. [11, 29] studied the performance of GDPR compliant systems. Their work mainly considers rights of data subjects (e.g., customers) and provide a benchmark to evaluate data processing aspects including metadata indexing, deletion, access control, and encryption. [19, 18] proposed an architectural vision for a database that natively supports auditing, deletion, and user consent management. [28] and [27] examine how GDPR affects the design and operation of modern computing systems. [24] and [34] studied supporting restrictions based on "purpose"-based access control. [26] presents an analysis on the impact of GDPR on storage systems. [15] presents a vision for Software-Defined Data Protection, for which they propose leveraging recent advances in Software-Defined Storage (e.g., FPGA-based key-value stores) to achieve compliance at the storage level. In the context of dataflow processing, [30] investigated supporting of data subject's privacy request (for access, deletion, and objection) by adopting causal snapshot consistency. [4] focuses on auditing GDPR compliance based on logs. All the above work can be seen as complementary to our work. While the aforementioned works focus more on data subject's rights, we focus more on the actual processing of the data wherein we consider regulations pertaining to the movement of data. Perhaps, closest to our work on compliance-aware query optimization is that of [23] and [9, 8]. While these works only focus on the operator placement problem based on privacy or user-specified constraints, we additionally consider rewriting queries by extending and interleaving query operators with data masking functions.

Federated Data Processing. Our work is also related to earlier works on multi-database query processing [17] and more recent work on cross-platform systems [3] and polystores [7]. It differs from the former in that we focus on heterogeneous compute infrastructures and from the latter in that we target geo-distributed environments and a facilitate a decentralized query execution.

6 Conclusion

Growing concerns on data privacy and usage preclude data transfers across national (or organizational) borders. It is therefore crucial that data processing in federated environments be compliant with data transfer regulations. In this paper, we have analyzed data transfer regulations from the perspective of GDPR and discussed key research challenges for including compliance aspects in federated data processing. We presented our approach for compliant geo-distributed data processing that focuses on relational workloads. We also discussed how query rewriting and optimization techniques must be extended to support data masking and outlined open problems and research directions to support workloads beyond relational workloads. We have advocated the need for decentrally executing queries and presented challenges and research directions to support data analytics over geo-distributed and heterogeneous compute infrastructures.

Acknowledgements This work is funded by the German Ministry for Education & Research as BIFOLD – "Berlin Institute for the Foundations of Learning and Data" (01IS18025A and 01IS18037A).

References

[1] Regulation (eu) 2016/679 of the European parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. 59:1–88, 2016.

[2] California consumer privacy act. California civil code, section 1798.100, June 28. 2018.

[3] Divy Agrawal, Sanjay Chawla, Bertty Contreras-Rojas, Ahmed Elmagarmid, Yasser Idris, Zoi Kaoudi, Sebastian Kruse, Ji Lucas, Essam Mansour, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, Saravanan Thirumuruganathan, and Anis Troudi. Rheem: Enabling cross-platform data processing: May the big data be with you! *Proc. VLDB Endow.*, 11 (11): 1414–1427, 2018.

[4] Debois S. Arfelt E., Basin D. Monitoring the GDPR. In *ESORICS*, pages 681–699, 2019.

[5] Kaustubh Beedkar, David Brekardin, Jorge-Anulfo Quiané-Ruiz, and Volker Markl. Compliant geo-distributed data processing in action. *Proc. VLDB Endow.*, 14 (12): 2843–2846, 2021.

[6] Kaustubh Beedkar, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. Compliant geo-distributed query processing. In *ACM SIGMOD*, page 181–193, 2021.

[7] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. The bigdawg polystore system. *SIGMOD Rec.*, 44 (2): 11–16, 2015.

[8] Nicholas L. Farnan, Adam J. Lee, Panos K. Chrysanthis, and Ting Yu. PAQO: A preference-aware query optimizer for postgresql. *Proc. VLDB Endow.*, 6 (12): 1334–1337, 2013.

[9] Nicholas L. Farnan, Adam J. Lee, Panos K. Chrysanthis, and Ting Yu. PAQO: Preference-aware query optimization for decentralized database systems. pages 424–435, 2014.

[10] GDPR Enforcement Tracker. `https://www.enforcementtracker.com/`.

[11] GDPRbench. `https://www.gdprbench.org/gdpr`.

[12] Goetz Graefe and William J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218, 1993.

[13] Fabian Hueske, Mathias Peters, Matthias J. Sax, Astrid Rheinländer, Rico Bergmann, Aljoscha Krettek, and Kostas Tzoumas. Opening the black boxes in data flow optimization. *Proc. VLDB Endow.*, 5 (11): 1256–1267, 2012.

[14] Chien-Chun Hung, Leana Golubchik, and Minlan Yu. Scheduling jobs across geo-distributed datacenters. In *SoCC*, page 111–124, 2015.

[15] Zsolt István, Soujanya Ponnapalli, and Vijay Chidambaram. Software-defined data protection: Low overhead policy compliance at the storage layer is within reach! *Proc. VLDB Endow.*, 14 (7): 1167–1174, 2021.

[16] Vanja Josifovski, Peter Schwarz, Laura Haas, and Eileen Lin. Garlic: A new flavor of federated query processing for db2. In *SIGMOD*, pages 524–532, 2002.

[17] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32 (4): 422–469, 2000.

[18] Tim Kraska, Michael Stonebraker, Michael Brodie, Sacha Servan-Schreiber, and Daniel Weitzner. *SchengenDB: A Data Protection Database Proposal*, pages 24–38. 2019.

[19] Tim Kraska, Michael Stonebraker, Michael Brodie, Sacha Servan-Schreiber, and Daniel J. Weitzner. Datumdb: A data protection database proposal. In *Poly'19 co-located at VLDB 2019*, 2019.

[20] Jayashree Mohan, Melissa Wasserman, and Vijay Chidambaram. Analyzing gdpr compliance through the lens of privacy policy. In Vijay Gadepally, Timothy Mattson, Michael Stonebraker, Fusheng Wang, Gang Luo, Yanhui Laing, and Alevtina Dubovitskaya, editors, *Poly'19 co-located at VLDB 2019*, 2019.

[21] Opinion 05/2014 on Anonymisation Techniques. `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`.

[22] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. In *SIGCOMM*, pages 421–434, 2015.

[23] Guido Salvaneschi, Mirko Köhler, Daniel Sokolowski, Philipp Haller, Sebastian Erdweg, and Mira Mezini. Language-integrated privacy-aware distributed queries. *Proc. ACM Program. Lang.*, 3 (OOPSLA), 2019.

[24] Malte Schwarzkopf, Eddie Kohler, M. Frans Kaashoek, and Robert Morris. Position: GDPR compliance by construction. In *Poly'19 co-located at VLDB 2019*, 2019.

[25] Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips, Wenlei Xie, Yutian Sun, Nezih Yegitbasi, Haozhun Jin, Eric Hwang, Nileema Shingte, and Christopher Berner. Presto: Sql on everything. In *ICDE*, pages 1802–1813, 2019.

[26] Aashaka Shah, Vinay Banakar, Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. Analyzing the impact of GDPR on storage systems. In *USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, 2019.

[27] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. The seven sins of Personal-Data processing systems under GDPR. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[28] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. Gdpr anti-patterns: How design and operation of modern cloud-scale systems conflict with gdpr, 2019.

[29] Supreeth Shastri, Vinay Banakar, Melissa Wasserman, Arun Kumar, and Vijay Chidambaram. Understanding and benchmarking the impact of gdpr on database systems. *Proc. VLDB Endow.*, 13 (7): 1064–1077, 2020.

[30] Jonas Spenger, Paris Carbone, and Philipp Haller. Wip: Pods: Privacy compliant scalable decentralized data services. In *Poly 2021 and DMAH 2021*, page 70–82, 2021.

[31] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, Paul Bardea, Amruta Ranade, Ben Darnell, Bram Gruneir, Justin Jaffray, Lucy Zhang, and Peter Mattis. Cockroachdb: The resilient geo-distributed sql database. In *ACM SIGMOD*, page 1493–1509, 2020.

[32] Jonas Traub, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. Agora: Bringing together datasets, algorithms, models and more in a unified ecosystem [vision]. *SIGMOD Rec.*, 49 (4): 6–11, 2021.

[33] Ashish Vulimiri, Carlo Curino, Brighten Godfrey, Konstantinos Karanasos, and George Varghese. Wanalytics: Analytics for a geo-distributed data-intensive world. In *CIDR*, 2015.

[34] Lun Wang, Joseph P. Near, Neel Somani, Peng Gao, Andrew Low, David Dao, and Dawn Song. Data capsule: A new paradigm for automatic compliance with data privacy regulations. *CoRR*, abs/1909.00077, 2019.